ABSTRACT
A terminal operating system (TOS) is essential for efficient and productive terminal operations, as it supports planning, scheduling and equipment control. Increasingly, functions in the TOS are automated. As scope and level of automation are increasing, it is crucial that the software is well tested and fine-tuned before putting it in live operation. The traditional ways of testing and tuning the terminal operating system, as well as training its users are limited, leading to unnecessary risks. In this paper, we present a meanwhile proven, safe and inexpensive approach to test and tune the terminal operating system and train operators on an emulated virtual terminal. This novel approach in the field of container terminals has been successfully applied at over twenty container terminals during the previous past years.

Keywords: simulation, emulation, container terminal, optimization

1 INTRODUCTION
A terminal operating system (TOS) is a software application supporting the planning, scheduling and equipment control activities of a container terminal and by this being responsible for accurate operations within the terminal (Agerschou et al., 2004), (Stahlbock and Voß, 2008). As such, it is the heart of terminal operations, making its reliability and ability to enable high performing operations of essence. Even short hick-ups can cause substantial financial damage to a terminal as one hour of operational downtime may cost in excess of $50,000 for a large container terminal.

At the same time, terminal operators are requesting more functionality from their TOS, not in the last place while they are searching for greater process automation, going away from traditional manual planning and dispatching practices. This added functionality leads to further complication of the software in itself, adding to the risk of instability and thus operational downtime. Therefore, it is not surprising that, despite the great effort of software providers and - effort of future users, today’s practice is that TOS implementations and upgrades can cause quite some limitations and performance issues. One of the problems we observed, is the way the systems are tested and commissioned is limited to isolated processes, whereas day-to-day practice involves all kind of real-time interactions between the system’s components and processes executed. Besides, we noticed that during the testing phase, there is little attention for the performance that the TOS enables. If scheduling and dispatching algorithms are not clever enough, performance may be limited by the control software, rather than by the physical equipment executing instructions.

As the last points become of eminent importance, they motivated us to investigate the possibility to consider a simulated container terminal environment and a real TOS allowing for off-line experimentation with the TOS under near-to-live scenarios, hence including nearly all relevant dynamic interactions and events that are experienced during live operations at a container terminal. Herewith a clear overview can be created in which parameters can be
fine-tuned during software development, commissioning, and during training and operations. As we could not find anything suitable in the market place at the time we initiated this activities, we developed an emulation tool – called CONTROLS (which stands for CONtainer TeRminal Optimised Logistics Simulation). In the meantime two other systems appeared in the market used for testing TOS using emulation: VITO (Virtual Terminal Optimization) and ChessCon from Institute of Shipping Economics and Logistics (Schütt, 2011).

This paper concerns the concept behind emulation, as well as its application to test, tune and train various TOS’s. In the first place we present the concept of emulation in general, followed by the specifics of emulation for container terminal. After that we discuss in more detail the added value and application of emulation for testing, tuning and training TOS’s. During the discussion we present some findings from several projects that have been carried out using emulation as a modern approach to perform testing and tuning the TOS, as well as training its (future) users. In the last section, we draw conclusions and provide an outlook for future development.

2 EMULATION OF CONTAINER TERMINAL

Emulation refers to the ability of a software application or physical device to imitate another software application or device. Emulation is also considered as a certain phase within testing process of a controlled system. The development of a complex system (e.g. a container terminal), which is controlled by a separate control system (e.g. a TOS), may include one or more of the following phases, which aim to test the system during different design stages (Auinger et al, 1999).

- Full simulation: includes the simulation of both the complex system and the control system;
- Real-time control: uses real complex system and simulates the control system;
- Emulation: simulates the complex system and uses real control system;
- Prototyping: involves tests with real complex system and real control system.

![Image of Prototyping vs. Emulation](image-url)

**Figure 1.** Prototyping vs. Emulation
While full prototyping seems the most realistic testing possibility, it is quite expensive to build and experiment with the whole prototype system, especially because it involves the risk of errors if the possibilities of its design are not tested thoroughly beforehand. Full simulation involves lower costs, however, it may disregard some phenomena that are present in the real system, or it may contain additional factors that might influence the outcomes. Emulation and real-time control have the advantage in that they can be carried out in a cheaper way than full prototyping, they stay closer to reality, and they are, therefore, less time-consuming than full simulation (Verbraeck et al, 2000), (Mueller, 2001), (Boer, 2005).

An emulation of a container terminal (a virtual representation of the container terminal’s physical processes, and related operator behaviour) acts as a real terminal, i.e. provides a valid representation of the physical processes at a terminal (equipment behaviour, driver behaviour, operational scenarios – gate arrivals, train arrivals, vessel arrivals) (Schütt, 2011). This can be linked to the TOS in such a way that the TOS treats the model as the “real world” (via existing interface between TOS and equipment), and it can be used to run operational scenarios, either as they occurred in the past or as configured by the user (see Figure 1).

By examining the performance of the terminal controlled by TOS under various emulated conditions, an assessment can be made of the system (TOS but also the operational processes) in its actual configuration. This assessment is made by actually running an operation as it would normally take place at a terminal as well, however, without moving containers in a physical way. Now, the TOS communicates with virtual machines, drivers, clerks and other peripheral systems, rather than real ones. The communication protocol is completely identical to the communication in real-life. The emulation model includes a representation of all relevant processes at the terminal, e.g. the lay-out (yard, rail terminal and quay cranes), a model of the equipment (kinematics, driver behaviour, routing, disturbances, and availability), and performance measurement functionalities. All performance relevant interactions that take place between the equipment at the terminal (including the gate and rail terminal) have to be defined and supported by the interface between the TOS and the emulation tool (Kassl et al, 2008).

The performance of a terminal is jointly determined by the proper functioning of the TOS and the ability of the operators to use it. Given these aspects, we decided to develop an emulation environment with the aim to provide support by:

- Allowing for realistic, comprehensive, safe and inexpensive testing of the TOS for both green-field sites and for existing sites, implementing a new TOS, or receiving an upgrade.
- Enabling off-line tuning of TOS parameters, by allowing forecasting simulation, replay animation, and long term projection simulation.
- Allowing for realistic, real-time training, in a completely operational setting, for individual operational staff or complete teams (so-called human in the loop).

In order to enable emulation for the above mentioned goals, testing, tuning and training, we developed the CONTROLS emulation tool (Boer and Saanen, 2008). CONTROLS supports various types of operations and various TOS’s. The tool consists of a number of modules that can be applied and configured for each terminal. Experimenting with CONTROLS typically involves three steps. The first step, is the definition of experiment scenarios. The second step is the execution of the scenarios in the combined TOS and emulated terminal environment. The last step is to analyse the results of the experiments for each scenario. In the next sections results of various case studies are presented, illustrating the added value of emulation approach using CONTROLS for testing, tuning and training terminal operating systems.
3 TESTING USING EMULATION

In order to ensure proper working of a TOS, vendors as well as terminals spend a huge amount of time testing it. The way this is done, we address as “traditional software testing”. Although, this is in many cases highly automated, it is typically limited to testing specific processes in isolation, and just running particular scripts. In addition, there is still quite a lot of manual testing going on.

<table>
<thead>
<tr>
<th>Table 1. Traditional software testing vs. Testing using dynamic emulation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scope of testing</strong></td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Way of testing</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Way of assessment</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Emulation does not replace traditional ways of software testing, but adds on, for complex testing (Auinger et al, 1999). In Table 1, the differences are listed. When a testing exercise is executed with emulation, the control software is always in the loop. Test experiments typically comprise of an entire day of operations, and cover all kind of processes. From obvious loading and discharge, to more complex operations such as twin-lift, tandem-lift and dual cycling. The full chain of processes between gate, rail and vessel are covered. Experiments can be repeated as long as it takes to complete them flawless, and with the required performance, both technically (i.e. response times), and functionally (productivity and service levels). In a typical case, 5 – 10 typical operations are stored (‘scenarios’), and used as set of experiments that are representative for the (existing or future) operation.

Past experiences clearly show that many bugs are still found after the regular test cycles of the TOS vendors. We have also the experience that complex bugs are more likely to be found using emulation, as the coverage of the test scenarios is much greater. Furthermore, emulation-based testing automates quite a substantial amount of manual test cases. With prudent upfront mapping of test cases to the emulation scenarios, it can save 40 – 60% of manual (repetitive) testing.

Figure 2. RMG terminal in CONTROLS side by side to the TOS (SPARCS N4)
In emulation projects using CONTROLS, the emulation tool has been mostly used to support the introduction of new TOS or to update an existing one. In these projects, emulation proved to be a very valuable tool, enabling the development team to detect and solve about 95% of the errors that typically are found after going live.

Furthermore, the visualisation accompanying the emulation, provided useful insights during the process (see Figure 2). While customers of a TOS usually have to wait until the system goes live, they were now able to operate a virtual operation long before the system went into real operation. This allowed future operators to be trained and also provided early feedback from the future users.

Finally, emulation enabled the team to stay focused on the performance of the terminal, or system. Where typically, the focus shifts from performance to “getting the things to work”, a continued focus could be kept on performance (think of quay crane productivity, equipment utilization, stacking efficiency, truck service, etc.) because CONTROLS allowed for this. In one particular instance, the final delivery by the TOS vendor only achieved about 60% of the required productivity level, as a result of errors, malfunctioning algorithms, and the need for fine-tuning parameters. The emulation allowed to detect the errors, and guide the fine tuning in a systematic way, with the resulting performance numbers as way to assess the impact of the solutions developed. Finally, the targeted performance was achieved, and later also reproduced in reality.

4 TUNING USING EMULATION

For basic installations of a TOS, the level of automation of decision-making by the TOS is limited. However, more and more, additional functionality is being added, to handle large-scale operations in which human decision-making becomes difficult, to increase the density of the yard (more throughput per hectare), and to improve equipment productivity. The modules that are being added typically contain a large number of parameters that are interrelated. Typically, the settings are configured and tested in live operations, which are by nature not suitable for finding satisfying or optimal configuration. Live operations are not repetitive, are highly affected by irregular events (e.g. break-downs) and human behaviours, and the measurement tools are also limited.

In order to tune the parameter settings and verify or optimize the algorithms, emulation can be used off-line using real operational scenarios. These scenarios are either real scenarios, meaning that it’s a replay of a past existing operation, or they are representative created scenarios, for instance for peak circumstances (e.g. full yard, 20% volume growth, or a full berth after a storm) or breakdown situations.

The execution of a scenario consists of an initialization step during which the user (one time) plans the operation. These plans (settings) can be stored, in order to be able to repeat this experiment without initializing it again. The duration of an experiment run depends on the TOS, the emulation tool is capable of running as long as the TOS controls the operation. Attended runs may last as long as the user wants, as the emulation behaves as the real world would behave. Unattended runs, however, depend on the capability of the TOS to run without an operator controlling it.

As such, a past operation can be run over and over again using various parameters, rules and equipment configurations, in order to determine what is best in particular situations. The more complex algorithms and software is deployed, and the more automation is being introduced (both from control perspective as well as equipment perspective), the more need for such an approach for emulation rises.

Through emulation the experiment runs may be started for various scenarios. The output of the experiment runs can be saved and is available for in-depth analysis and reporting tools. The TOS performance can be assessed by comparing the performance of various simulated
operations, such as QC productivity, equipment productivity and status, truck and rail service times, number of shuffle moves, etc.

Emulation can be used either before implementing new software, focusing on the contribution to performance, or during operations, as a continuous improvement tool. As circumstances change, parameter settings in the software may (and will) also need to change to accommodate the increased yard density, equipment availability or gate peaks. In the next two subsections we present two tuning cases when emulation (CONTROLS) provided support to improve the performance of the TOS.

4.1 Improve truck turn-around time

In two of the cases (so far), we applied the emulation to improve productivity in a live environment, replaying past operations, evaluating different parameter settings. In one case, we aimed at improving truck turn-around time by improving straddle carrier dispatching. It appeared from the experiments that the truck turn-around time during the peak – by just changing the dispatching algorithm and the decision factors – could be reduced by more than 30% on average, and the peaks by even 50%. At minimal investment – it implied some changes to the TOS – the service to trucks could be improved drastically. In other words, the number of straddle carriers required to handle a certain truck peak could be reduced, saving labour and equipment costs. Figure 3 depicts the turn time and straddle productivity from this project. The optimized settings determined using emulation were implemented in August 2010. Since then, turn time has been averaging around 37 minutes, whereas 7 months before, they averaged at 49 minutes. This means a decrease of 25% of the turn time. At the same time, straddle productivity increased from average of 2.4 trucks per hour to 4.1 trucks/hour, an increase of 70% thus. Note that January and February 2011 were bad months due to excessive snow fall.

![Figure 3. Turn time and straddle productivity (in trucks handled per hour)](image-url)
4.2 Improve the yard planning strategy

Most recently, we carried out some studies for an RTG terminal that uses SPARCS terminal operating system. The goal of the study was to investigate the possibility to replace the currently applied yard planning strategy (based on the use of pre-stacks) with controlled random stacking strategy.

Proper yard planning strategies help to assign the containers to an optimal position in the yard. As a result of this, the re-handle moves and yard shifts can decrease, and the yard utilization and productivity can increase. In theory there exist different strategies, such as pre-marshalling (Chen, 1999), (Lee and Hsu, 2007), (Caserta et al, 2011), sort and store (Kim and Kim, 1999), (Kim and Park, 2003) controlled random strategy (Duinkerken et al, 2001) (Dekker et al, 2006), etc.

We defined different scenarios, in each one we modified the expert decking parameters according to certain aspects, such as: the workload of RTGs (e.g. increase/decrease the influence of RTG related variables), the travelling distance of TTs (e.g. increase/decrease the value of penalties related to terminal truck driving distance), specific yard settings (e.g. impossible to stack containers on top of containers that have a different type or which are planned to be moved). For each scenario, we carried out experiments and investigated which aspects are the most relevant. We concluded that with proper settings of the parameters the controlled random stacking strategy indeed can be a good choice as it improves the quay crane and RTG productivity.

![Figure 4. Comparing QC Net productivity applying different yard strategies](image)

We achieved significant improvements (5-10% increase) of quay crane productivity applying the SPARCS expert decking functionality (see Figure 4). We realized this by changing the grounding parameters (for instance allocation filters in combination with equipment control parameters, the weight factor of travel distance, etc.).
5 TRAINING USING EMULATION

As operations become larger, denser and faster, and TOS contain more functionality to control these operations, the interaction between the TOS and the user (e.g. vessel planner, yard planner, equipment dispatcher) becomes more complex. This in the sense that there is more to decide for the user, the decisions have effects over long periods of time (up to weeks), and wrong decisions have a higher financial impact. Therefore, it’s crucial that TOS operators (to address all its users) know how the TOS works, and how they should handle irregularities and events that the TOS cannot handle automatically. This starts from simple “fail to deck” events, reacting to breakdown events and rescheduling work, to vessels arriving off-window, and deciding where to berth them, so that the cascading effect on other vessels is minimal, whilst still allocating a reasonable berth to the vessel in question.

Training of TOS operators is typically performed on the job, which means that a newcomer sits besides an experienced person for some time, and then is deployed on its own. The likeness of being capable then of handling less frequent but high impact events is small, which bares the risk of wrong decision-making. Besides, on the job training is typically not followed by any standardized form of evaluation to ensure that the newcomer is now able to handle the situations he will be faced with during operations. In addition, during his work, there is little evaluation of his work either, i.e. there is typically no feedback loop from work (e.g. planning) done in the past to the actual realized performance. As such, inadequate capabilities may remain unnoticed for longer periods of time.

In the emulation-based training, a real operation is being emulated in real-time, and the TOS user is fulfilling his role as part of the operational team. The TOS user fulfils his regular tasks, using the TOS, making his decisions, with the only difference to reality that his work leads to virtual container movements. Using emulation, “difficult” scenarios can be configured and events can be prepared (e.g. breakdowns, changing information, drivers that arrive in the wrong sequence) that test the capabilities of the TOS user. As such, the TOS user is trained in an environment that allows focusing on all tasks under regular and irregular circumstances, without affecting live operations.

An emulation-supported training that we have conducted, involving 6 vessel planners clearly shows the impact of a good plan. Each of the 6 planners was targeted to plan the same vessel. The vessel was consequently executed after completion of plan, all against an emulation of exactly the same operation, i.e. the same amount of equipment, the same initial yard, the same behaviour of equipment. The results are shown in Figure 5; it shows for each vessel planner the average crane productivity (in bx/h) as well as the vessel turn time (in hours). Note that a higher crane productivity does not always mean a shorter turn time, as work distribution among the cranes (up to 4 cranes were deployed per vessel) determines how long it takes to handle a vessel.

---

1 The inability of the TOS to place a container fully automated in the storage yard, due to a lack of space within the user-defined allocation ranges.
Three vessel planners were requested to use the provided automated stowage planning module available in the TOS. Despite their lacking experience with this tool, they all performed better by using it. Unfortunately, the group is too small to say anything statistically sound about the contribution of the automated stowage planning, however, the results as shown in Figure 5 give a clear indication that it is substantial. The average turn time (see the column in hours, varying from 10.3 hours till as high as 15.7 hours) decrease with 18%, whereas the average crane productivity increased with 26%. Furthermore, all planners that used the automated stowage planning turned the vessel quicker than the ones that practiced common procedures, and on top needed 25% less time to complete the planning process. Moreover, we can say that this way of training allows objective measurement, and safe try out of new methods, in this case for vessel planning. The case studies clearly show that the presented emulation approach indeed provides a safer and cheaper way to test and tweak the TOS and train operators on an emulated virtual terminal.

6 CONCLUSION

Due to the continuous volume increase, the need of handling more and more containers within given time window, and a limited yard place; larger vessels, new generation equipments, bigger terminals and more optimized operations are considered (Saanen, 2004). Since a container terminal’s performance relies on its TOS, it is vital to continuously improve the capability of the TOS. As a TOS gets more advanced and as terminals get more reliant on their TOS, the importance of sophisticated test, tuning and tweaking tools gets more vivid. Simply allowing TOS vendors to deliver new releases without running realistic scenarios, installing TOS without detailed performance testing during the commissioning, and developing TOS without the use of tools that make the performance metrics explicit, is something that can entail a huge risk and expenses.

In order to reduce the risk and expenses, an emulation tool has been developed, called CONTROLS. Emulation allows the user to experiment with the real TOS without the risk of affecting (negatively) real operations. Problems caused by the TOS can be recognized
immediately and solved before the software is put live. With CONTROLS, we have taken container terminal emulation to a mature point, offering a tool that can complement live operations with a valid, detailed, and advanced laboratory environment for testing, tuning and training. CONTROLS has proven itself in the various applications at the terminal – meanwhile we completed more than 20 projects -, by finding bugs, malfunctioning algorithms, allowing for tuning parameter settings as well as training operations staff in a systematic and measureable way.

The next challenge is to make this way of training the standard, as flight simulators are for pilots. So far the experience with the human in the loop, limited to three cases, and adoption goes slow. As this approach deviates substantially from today’s way of training, using the emulation as training environment is a big step.

In addition, tuning algorithms and settings that are related to grounding containers, require experiments with a long duration. Here, today’s TOS are not supportive in the sense that almost continuously (and at least every shift) user input is required to keep the operation – also in emulation mode – going. For tuning purposes, this is quite labour intensive, which is a potential bottleneck for extensive experimentation. We intend working on simulated TOS users to overcome that matter, but there is a long way to go, as the information that they use, is much richer than what the TOS contains, i.e. decisions of users are based on additional information as well (received by various media for instance).

Emulation allows not only experimenting with existing terminals but also with the future terminals. Accordingly, emulation also provides terminals with the opportunity to increase throughput without adding equipment or yard space: simply by doing more with less. As volumes increase, costs rise, and space is scarce, emulation provides a very promising way out.

ACKNOWLEDGEMENTS

We would like to thank our colleagues for their support in developing and testing CONTROLS.

REFERENCES


